Visualization and Analysis of HPC Simulation Data with VisIt and Ascent

ATPESC 2026

Data Analysis and Visualization Track

2026/07/29

Cyrus Harrison

LLNL

Prepared by LLNL under Contract DE-AC52-07NA27344.

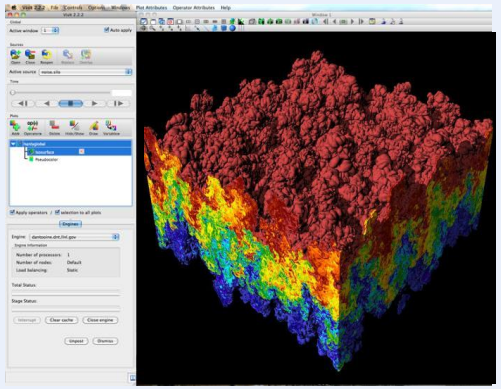
Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

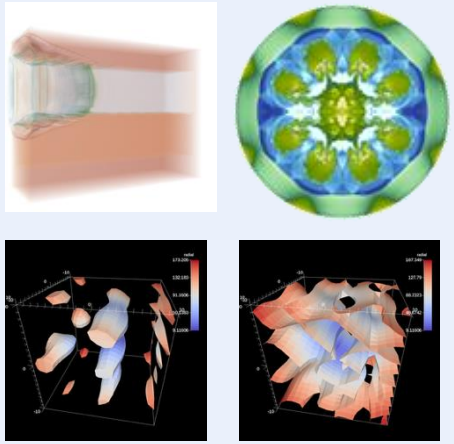
The VisIt team develops open-source Visualization, Analysis, and I/O tools

VisIt



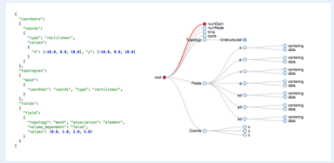
Turnkey HPC application for visualization and analysis of simulation data

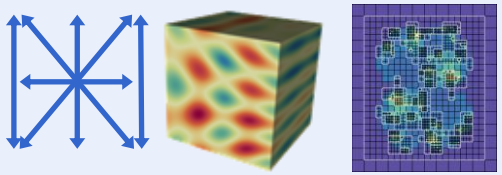
Ascent



Easy-to-use flyweight in situ visualization and analysis library for HPC simulations

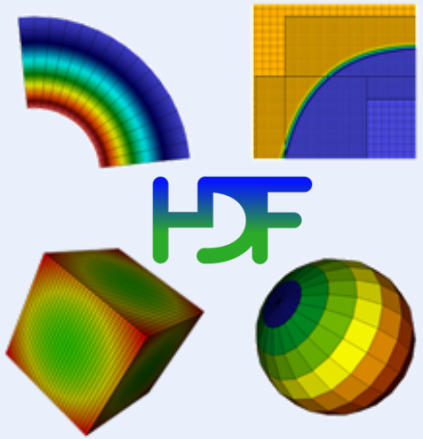
CONDUIT





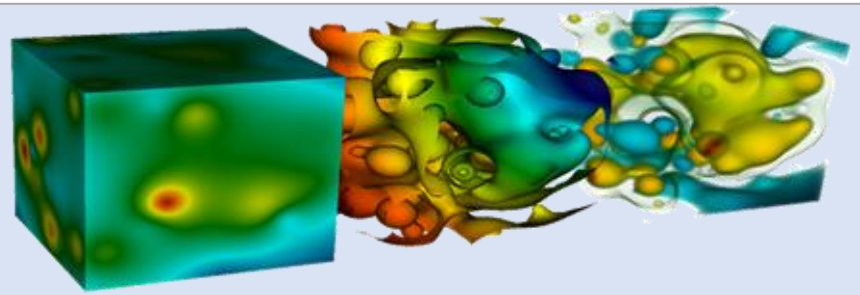
In-memory data description, HPC I/O, and shared schemas for simulation data exchange

Silo

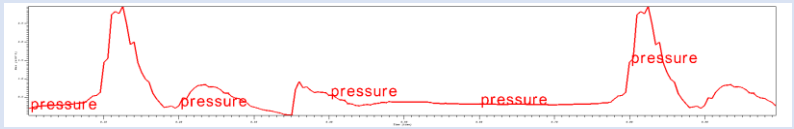


File-based, scientific data exchange library for checkpoint restart and visualization

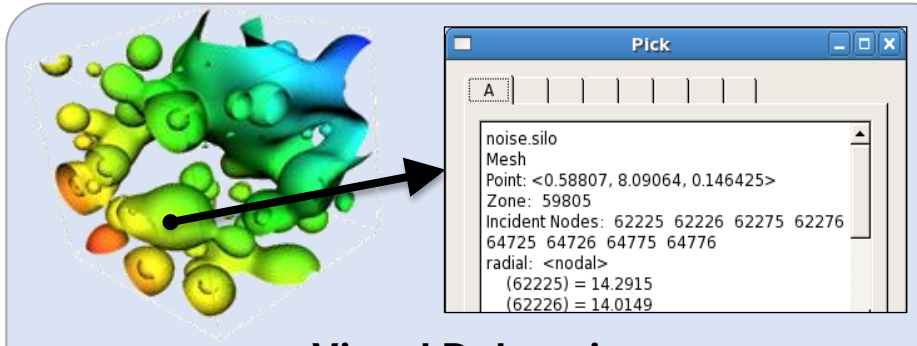
Scientific visualization tools support a wide range of use cases



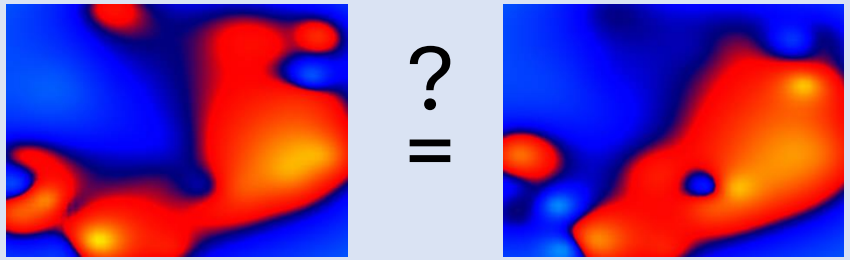
Data Exploration



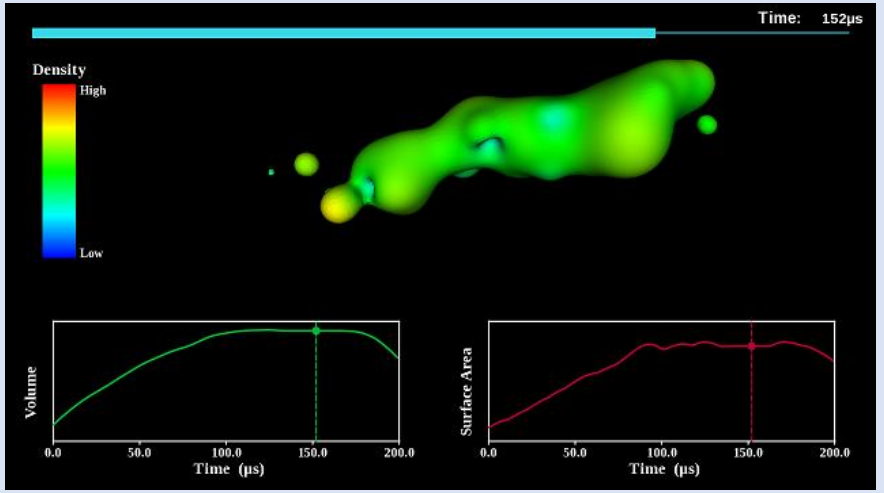
Quantitative Analysis



Visual Debugging

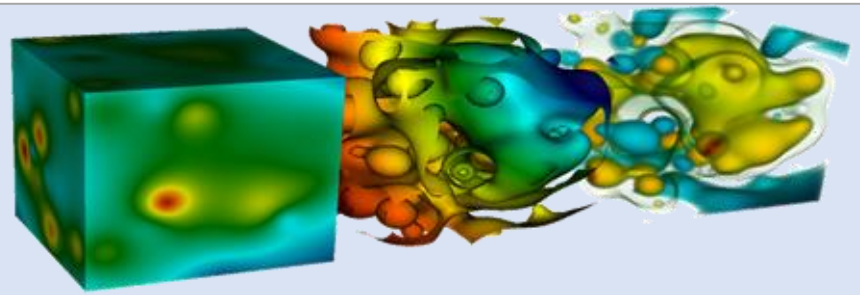


Comparative Analysis

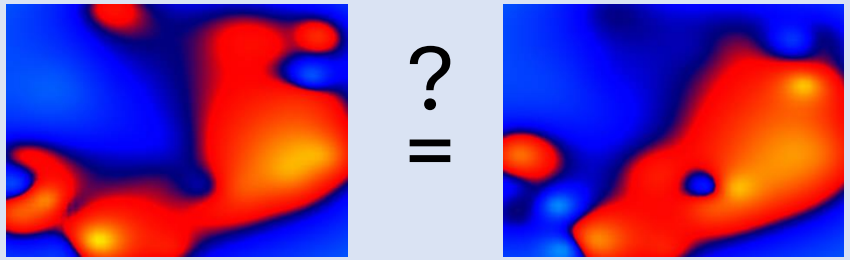


Presentation Graphics

Scientific visualization tools support a wide range of use cases

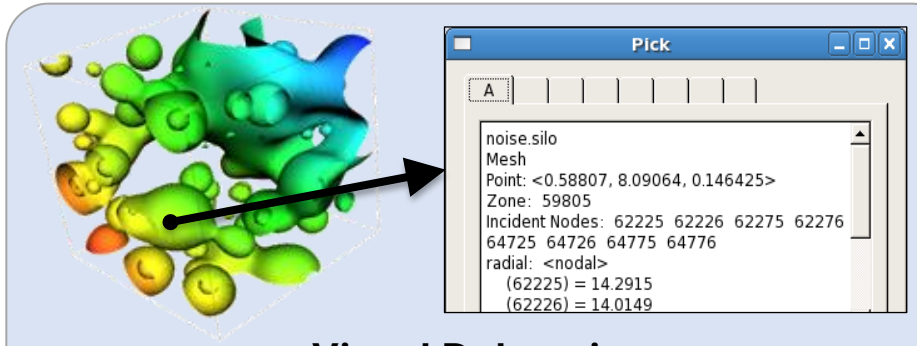


Data Exploration

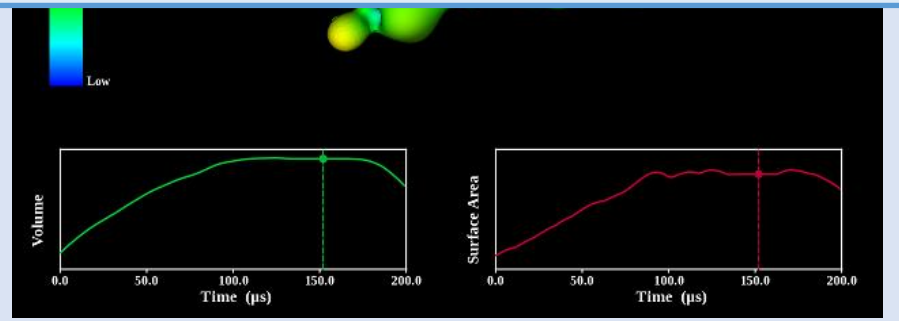


Comparative Analysis

These tools are used daily by scientists to digest and understand HPC simulation results



Visual Debugging



Presentation Graphics

Scientific visualization tools are used both post hoc and in situ

Post Hoc

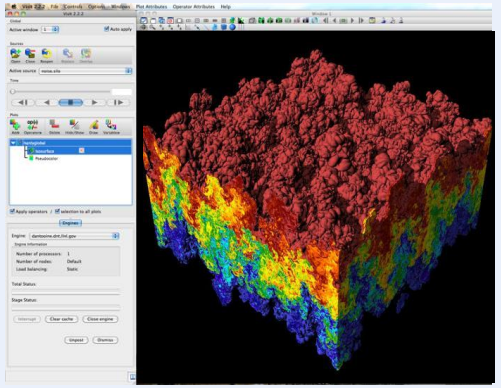
Simulation data is processed after the simulation is run using distinct compute resources.

In Situ

Simulation data is processed while it is generated, sharing compute resources with the simulation application.

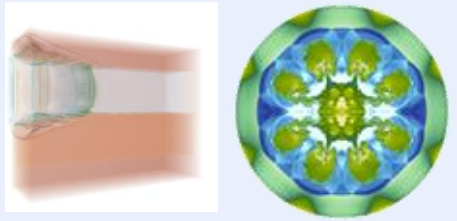
This Tutorial will give you a brief intro to both VisIt and Ascent

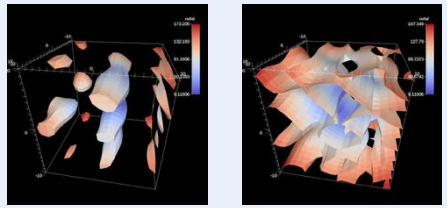
VisIt



Turnkey HPC application for visualization and analysis of simulation data

Ascent

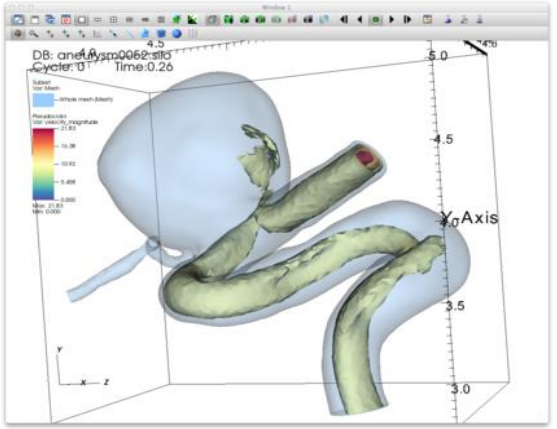




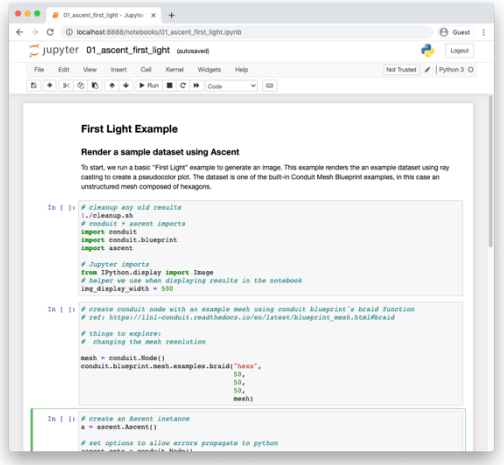
Easy-to-use flyweight in situ visualization and analysis library for HPC simulations

Tutorial Outline

- **VisIt**
  - Project Introduction (10 min)
  - Visualization of an Aneurysm (30 min) (Blood Flow) Simulation
- **Ascent**
  - Project Introduction (10 min)
  - Tutorial Notebooks (10 min)



Simulation Exploration with VisIt



Ascent Tutorial Notebook

Tutorial Resources

- **Tutorial Materials**
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/index.html
  - https://www.ascent-dav.org/tutorial/
- **VisIt**
  - VisIt 3.5.0 binaries: https://github.com/visit-dav/visit/releases
  - Tutorial Data: https://visit-dav.github.io/largedata/datarchives.html
  - GitHub: https://github.com/visit-dav/visit
  - GitHub Discussions: https://github.com/visit-dav/visit/discussions
- **Ascent**
  - ascent@develop
  - GitHub: https://github.com/alpine-dav/ascent

Tutorial Data Acknowledgements

- **Aneurysm Simulation Dataset**
  - Simulated using the LifeV finite element solver
  - **Available thanks to:**
    - Gilles Fourestey and Jean Favre
      Swiss National Supercomputing Centre (http://www.cscs.ch/)
- **Potential Flow Simulation Dataset**
  - Simple tutorial simulation built using MFEM (https://mfem.org/)
  - **Available thanks to:**
    - Aaron Fisher and Mark Miller, LLNL

VisIt Project Introduction

vis·it

/ˈvizit/

verb

1. go to see and spend time with (someone) socially.
   "I came to visit my grandmother"
   synonyms: call on, call in on, pay a call on, pay a visit to, pay someone a call, pay someone a visit, go to see, come to see, look in on; More

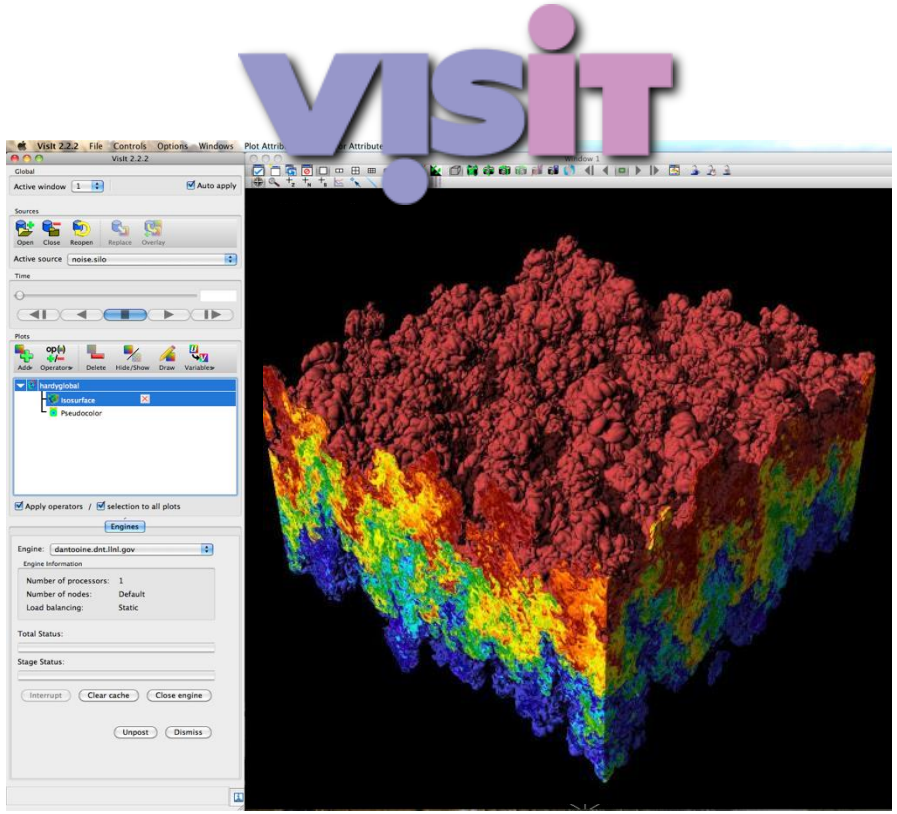
2. inflict (something harmful or unpleasant) on someone.
   "the mockery **visited upon** him by his schoolmates"
   synonyms: happen to, overtake, befall, come upon, fall upon, hit, strike
   "it is hard to imagine a greater psychological cruelty visited on a child"

noun

1. an act of going or coming to see a person or place socially, as a tourist, or for some other purpose.
   "a visit to the doctor"
   synonyms: social call, call
   "after reading the play she paid a visit to the poet"

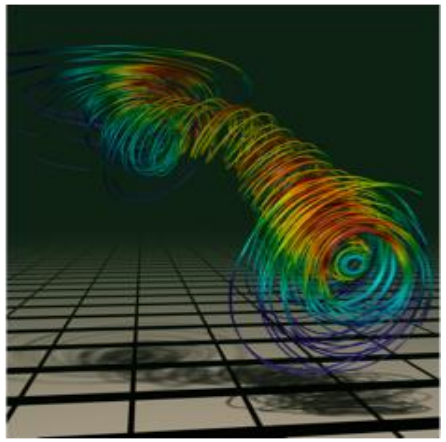
VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data

- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.

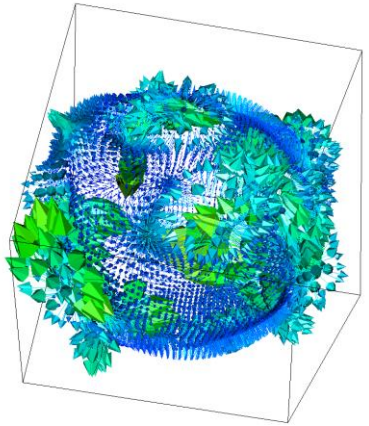


Pseudocolor plot of Density
(27 billion element dataset)

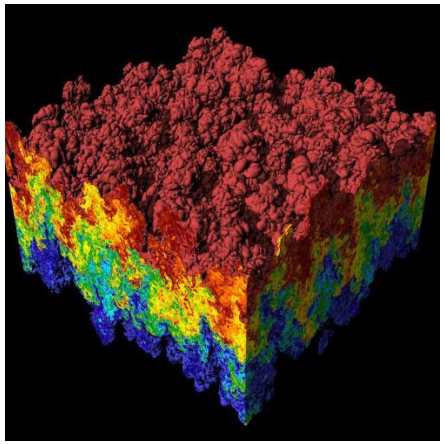
VisIt provides a wide range of plotting features for simulation data across many scientific domains



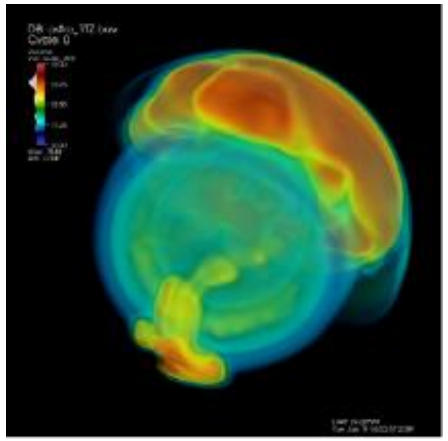
Streamlines / Pathlines



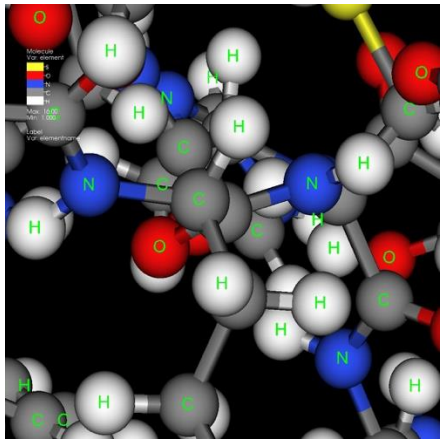
Vector / Tensor Glyphs



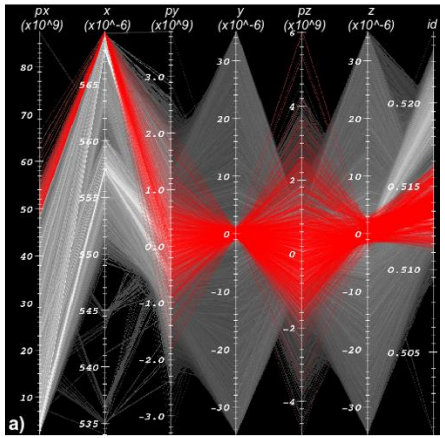
Pseudocolor Rendering



Volume Rendering



Molecular Visualization



Parallel Coordinates

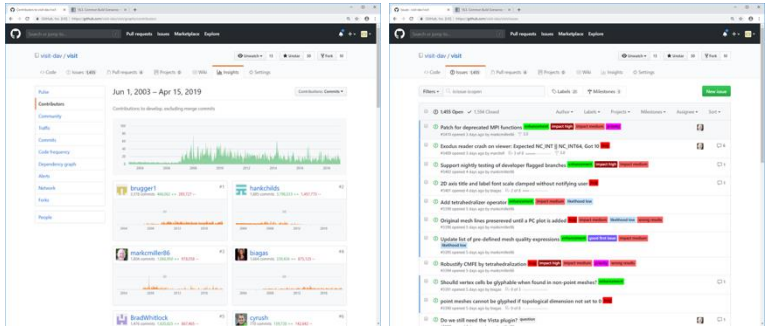
VisIt is a vibrant project with many participants

- The VisIt project started in 2000 to support LLNL's large-scale ASC physics codes
- The project grew beyond LLNL and ASC with development from DOE SciDAC and other efforts
- VisIt has had contributors from a wide set of organizations:
  - LLNL, LBNL, ORNL, Univ of Oregon, Univ of Utah, Intelligent Light, …
- Over 200 person years of effort, 2+ million lines of code
- VisIt has hundreds of active users at LLNL and thousands worldwide

VisIt is hosted and developed using GitHub

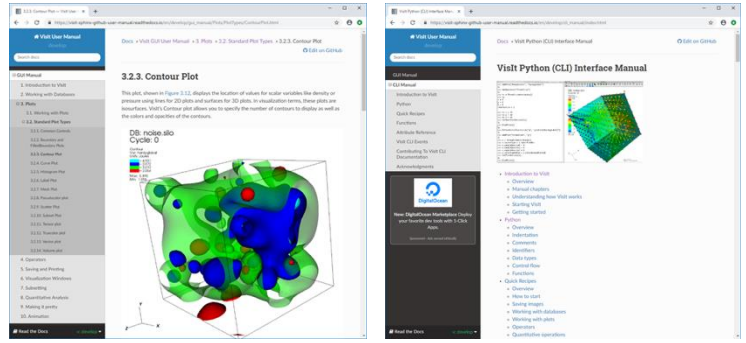
- Main Website
  - https://visit-dav.github.io/visit-website
- Our Source Code, Issue tracking, and Discussions are in the `visit-dav` GitHub organization
  - https://github.com/visit-dav/
- Our Docs are hosted on Read the Docs
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/

GitHub



VisIt source repo and issue tracking on GitHub

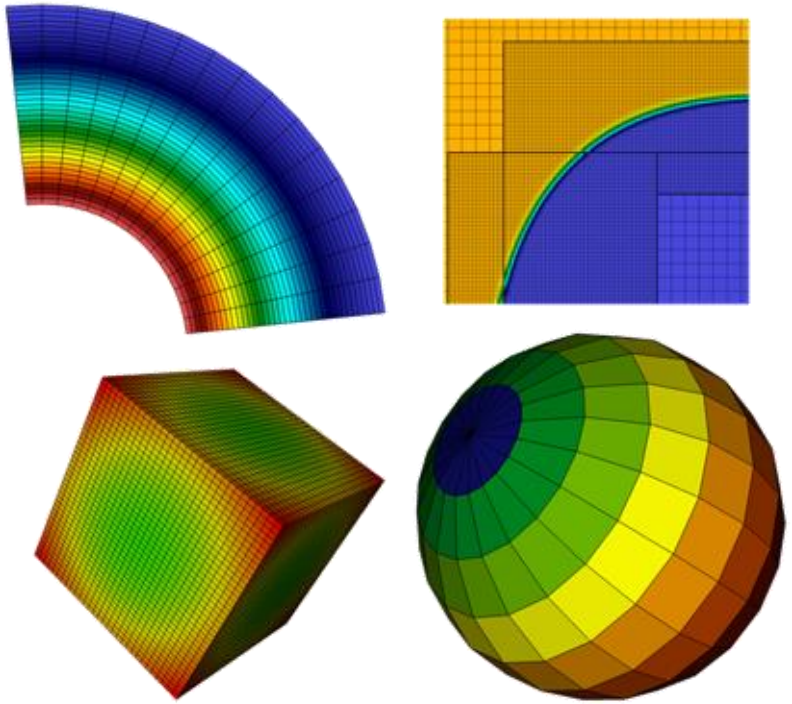
Read the Docs



VisIt manuals on Read the Docs

VisIt provides a flexible data model, suitable for many application domains

- **Mesh Types**
  - Point, Curve, 2D/3D Rectilinear, Curvilinear, Unstructured
  - Domain Decomposed, AMR
  - Time Varying
  - Primarily linear element support, limited quadratic element support
- **Field Types**
  - Scalar, Vector, Tensor, Material Volume Fractions, Species



The VisIt team releases binaries for several platforms and a script that automates the build process

“How do I obtain VisIt?”

- Use an existing build:
  - For your Laptop or Workstation:
    - Binaries for Windows, OSX, and Linux (RHEL, Ubuntu, and many other flavors): (https://github.com/visit-dav/visit/releases/)
  - Several HPC centers have VisIt installed
- Build VisIt yourself:
  - “build_visit” is a script that automates the process of building VisIt and its third-party dependencies. (also at: https://github.com/visit-dav/visit/releases/)
  - Fledgling support for building via spack (https://github.com/spack/spack)

VisIt supports more than 110 file formats

“How do get my data into VisIt?”

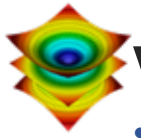
- The *PlainText* database reader can read simple text files (CSV, etc)
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/data_into_visit/PlainTextFormat.html
- Write to a commonly used format:
  - *VTK, Silo, Xdmf, PVTK, Conduit Blueprint (JSON/YAML, or HDF5 files)*
- We are investing heavily in Conduit Blueprint Support
  - http://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html
- Consult the Getting Data Into VisIt Manual

We continuously evolve our software development processes and resources

Overview of continuous technology refresh (CTR) on VisIt

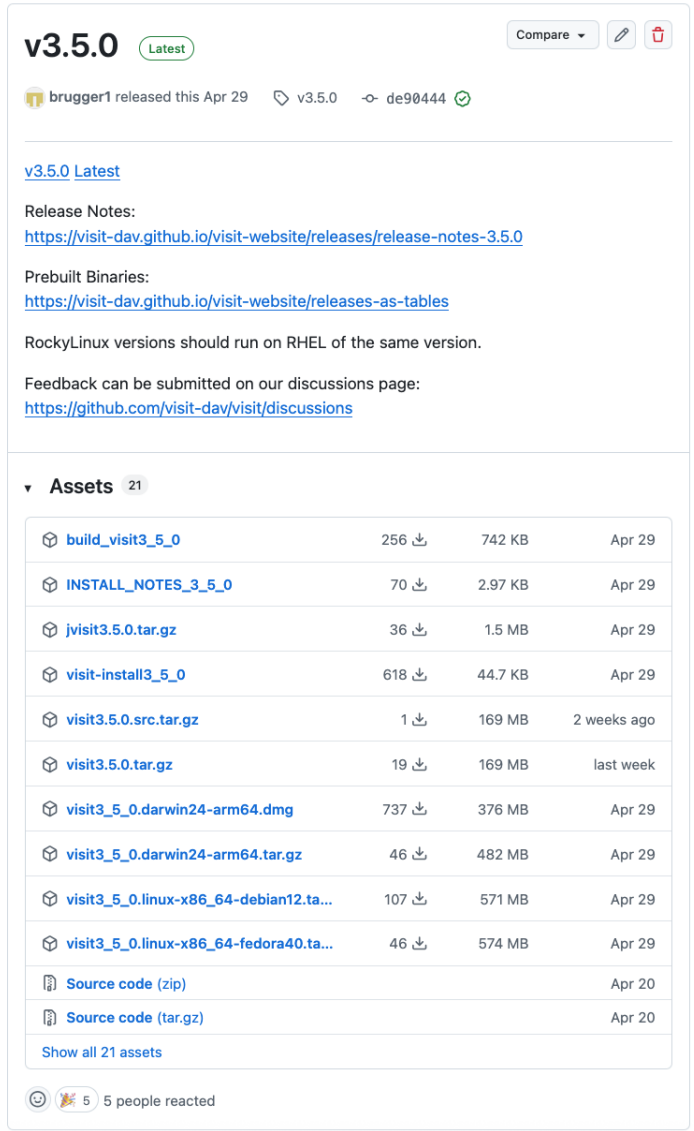
	1999 (LLNL Internal)	2008 (NERSC)	2019 (GitHub)	Notes
Revision control	ClearCase (LLNL/Yellow)	Subversion (NERSC)	Git (GitHub)	Binary content, git-lfs, svn->git full history, custom scripts
Issue Tracking	ClearQuest (LLNL/Yellow)	Redmine (ORNL)	Issues (GitHub)	Tied to code
Testing+Dashboard	B-Div Irix (LLNL/Yellow)	LLNL-CZ+ (NERSC)	LLNL-CZ+ (GitHub)	3k image+2k txt, fix/rebase tests, exact vs. fuzzy match
CI Testing	N/A		Cicle-CI → Azure	Presently ensuring only compile of core
User contact	Majordomo (LLNL)	GNU Mailman (ORNL) 4-2Viz (LLNL)	Discussions (GitHub) 4-2Viz, Teams (LLNL)	Discoverable, attachments/size, notification controls Privacy, where users are hanging out
Documentation	FrameMaker	OpenOffice	Sphinx (ReadTheDocs)	Mergeable, committed & versioned w/code (docs like code)
Website	N/A	Drupal (LLNL, WSC web)	Jekyll + GH Pages	Developers can edit directly, GitHub Pages
Configuration	AutoTools	CMake		Native windows dev
Operating System • Windows • OSX/macOS • Linux	• XP Vista 7 8 9 10 11 • OSX-10.? 10.2 10.4 10.5/6 10.8 10.10 10.12 10.14 11 12 13 • RedHat +ubuntu +(fedora, debian, centos)			• Visual Studio, sys-call changes, manually trigger tests • Security changes getting harder to manage • No means to test variants fully
Core 3rd Party Libs • Qt • VTK • GL	• Qt3 Qt4 Qt5 Qt6 • VTK-5.0 VTK-5.8 VTK-6 VTK-8 VTK-9 • GL Drivers + Mesa (OpenGL, GL rendering changes)			**Qt+VTK+GL is a complex interdependency** • Integration w/GL tricky, no automated testing for GUI • API changes, baselines change • hw GL when possible, driver compat, baselines change
Language Standards • C++ • Python	• C w/classes templates OK C++ 11 allowed C++ 14 Required • Python 2 Python 2 or 3			• Very conservative in adopting new language features • A lot of users still using Python 2 workflows

We released VisIt 3.5.0 in April 2026

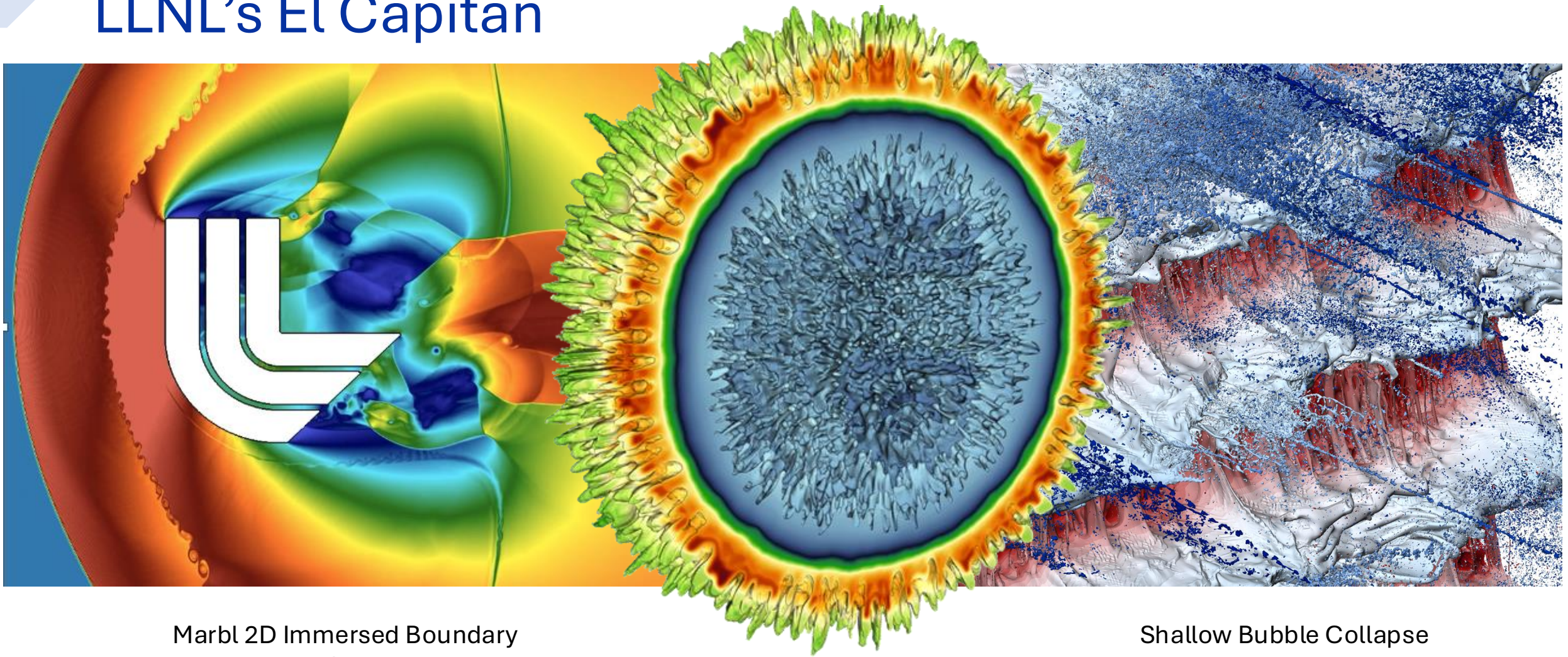


VisIt 3.5.0

- 50+ bug fixes, 50+ enhancements
- Major VTK update, new (simplified) GL strategy, ANARI support
- Many quality-of-life fixes for both overt and subtle issues
  (ex: high dpi support, memory leaks)
- Updated to use HDF5 2.0, and to use a single MPI-enabled HDF5 instead of separate serial/parallel libs
- Revamped MFEM LOR support
  (including H-div and H-curl cases, support for Quadrature Points)
- Blueprint support updates
- X Ray Image Query updates
- Ghost zone exchange updates
- Updated to Python 3.13, dropped Python 2 support, refactored how to import VisIt's Python module
- Build hardening for El Cap, TOSS4, Crossroads, and macOS
- We increased scope in fall: Added HDF5 2.0 target and MFEM features
  - Delayed release, but deemed by team too important to skip for 3.5.0
    (Minor release [X.*Y*] allows RPC VisIt protocol changes)
- Expanded use of Docker builds for releases on a wide set of Linux platforms, inc. RHEL
  https://visit-dav.github.io/visit-website/releases/release-notes-3.5.0/



VisIt is a key tool for users running simulations on LLNL's El Capitan

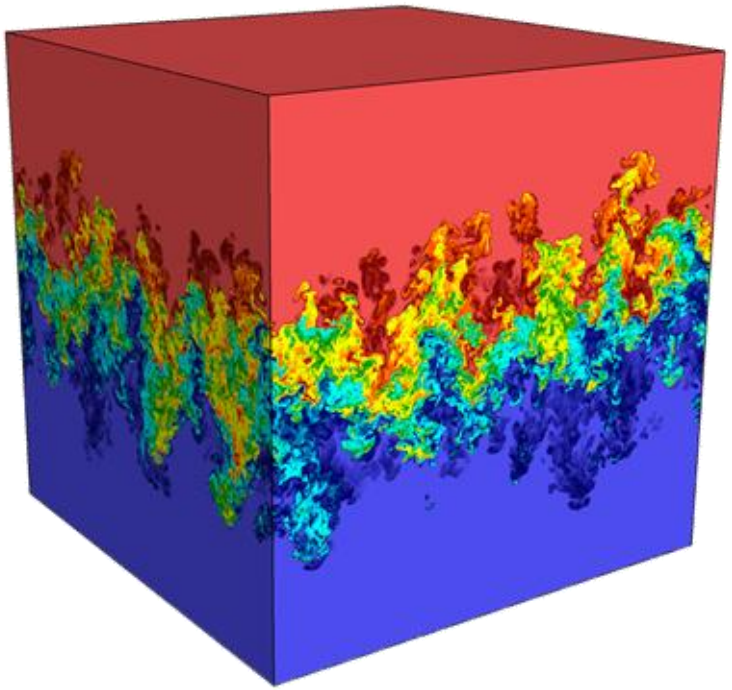


Marbl 2D Immersed Boundary
(B. Olson)

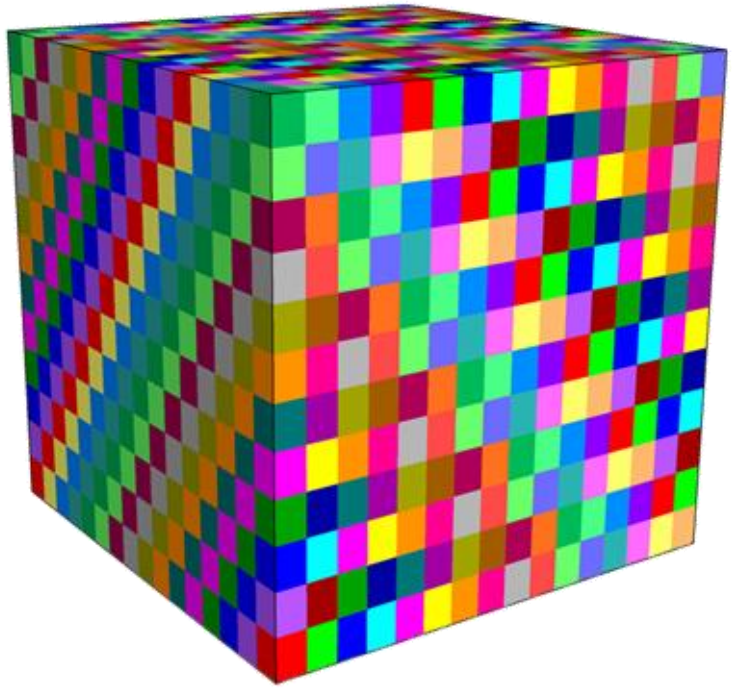
Marbl NIF N210808 Shot
(T. Stitt and R. Rieben)

Shallow Bubble Collapse
(J. Burmark, K. Mackay)

VisIt uses MPI for distributed-memory parallelism on HPC clusters

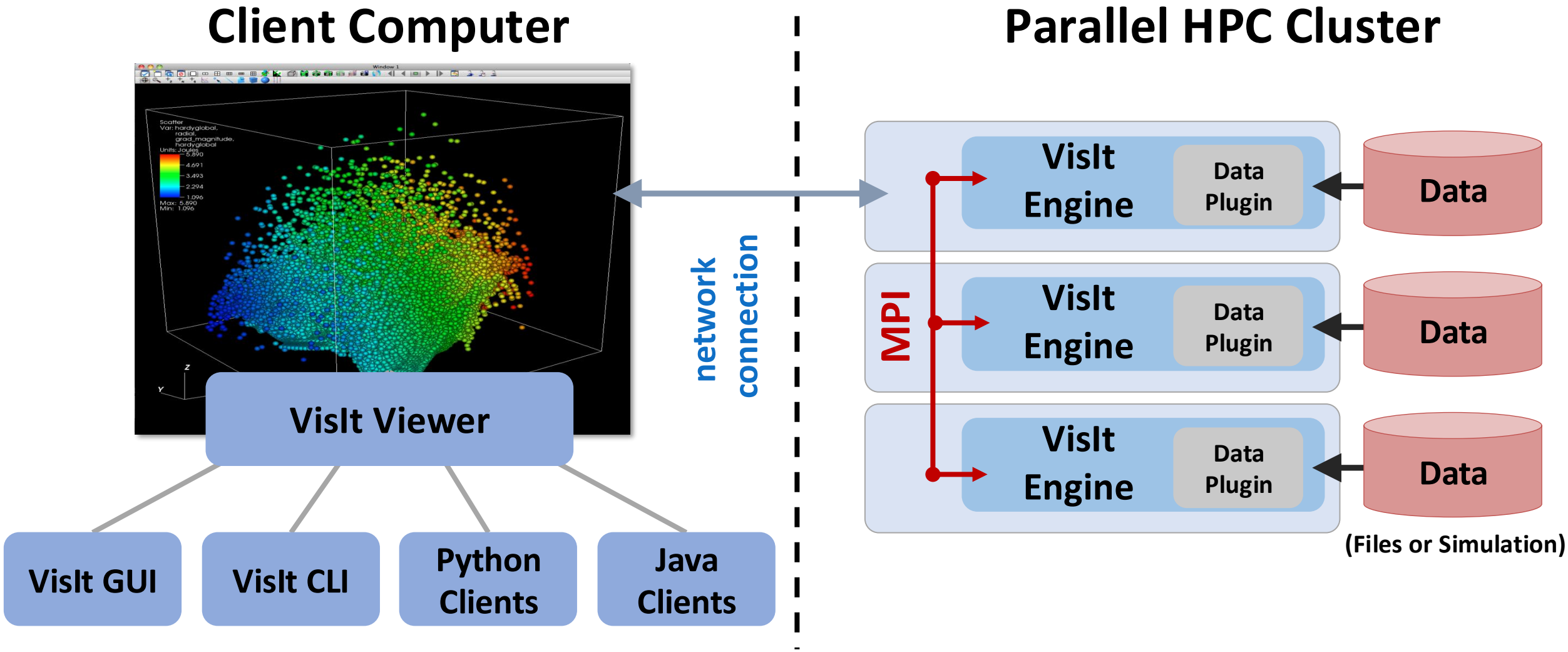


Full Dataset
(27 billion total elements)

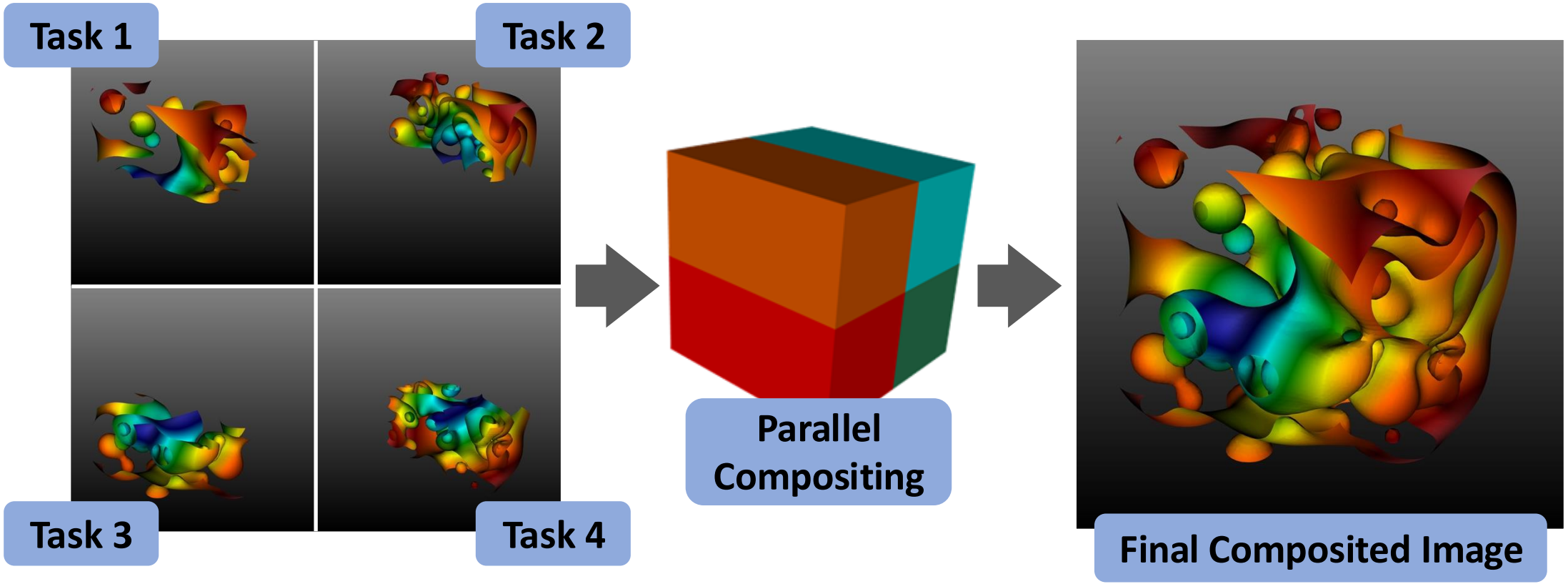


3072 sub-grids
(each 192x129x256 cells)

VisIt employs a parallelized client-server architecture



VisIt automatically switches to a scalable rendering mode when plotting large data sets on HPC clusters



In addition to scalable surface rendering, VisIt also provides scalable volume rendering

DOE's visualization community is collaborating to create open source tools for Exascale simulations

Addressing node-level parallelism

- Viskores (formerly VTK-m) is an effort to provide a toolkit of visualization algorithms that leverage emerging node-level HPC architectures from NVIDIA, AMD, Intel.

VISKORES

https://github.com/Viskores/

Addressing I/O gaps with in-situ

- Projects providing in-situ infrastructure and capabilities

CONDUIT

https://github.com/llnl/conduit

Ascent

https://github.com/Alpine-DAV/ascent

ParaView Catalyst

https://kitware.github.io/paraview-catalyst/

The VisIt team is investing in Conduit and Ascent to create next generation in situ infrastructure

Intuitive APIs for in-memory data description and exchange

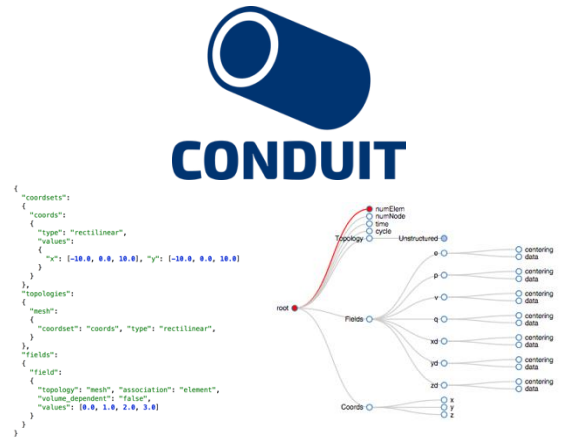
https://github.com/llnl/conduit

Flyweight in-situ visualization and analysis for HPC simulations

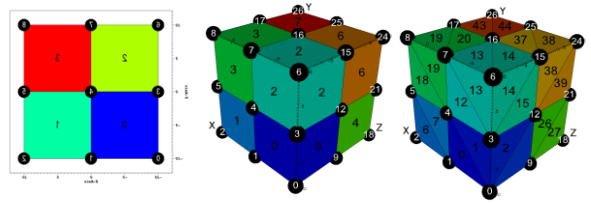
https://github.com/Alpine-DAV/ascent

Conduit provides intuitive APIs for in-memory data description and exchange

- **Provides an intuitive API for in-memory data description**
  - Enables *human-friendly* hierarchical data organization
  - Can describe in-memory arrays without copying
  - Provides C++, C, Python, and Fortran APIs
- **Provides common conventions for exchanging complex data**
  - Shared conventions for passing complex data (e.g. *Simulation Meshes*) enable modular interfaces across software libraries and simulation applications
- **Provides easy to use I/O interfaces for moving and storing data**
  - Enables use cases like binary checkpoint restart
  - Supports moving complex data with MPI (serialization)



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

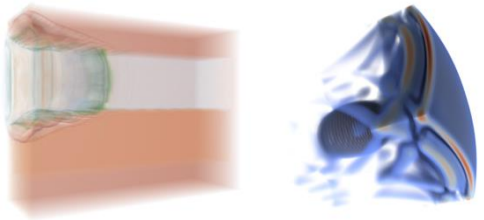
http://software.llnl.gov/conduit
http://github.com/llnl/conduit

Website and GitHub Repo

Ascent is an easy-to-use flyweight in situ visualization and analysis library for HPC simulations

- **Easy to use in-memory visualization and analysis**
  - Use cases: ***Making Pictures, Transforming Data,*** and ***Capturing Data***
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
  - Efficient distributed-memory (MPI) and many-core (CUDA, HIP, OpenMP) execution
    - Demonstrated scaling:
      In situ filtering and ray tracing across ***16,384 GPUs*** on LLNL's Sierra Cluster
  - Has lower memory requirements than current tools
  - Requires fewer dependencies than current tools (ex: no OpenGL)

Ascent



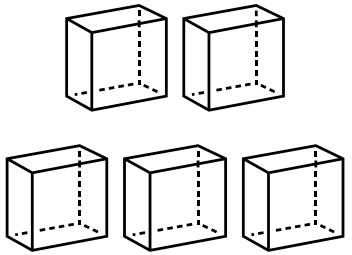
Visualizations created using Ascent

HDF python™ CINEMA

Extracts supported by Ascent

http://ascent-dav.org
https://github.com/Alpine-DAV/ascent

Website and GitHub Repo



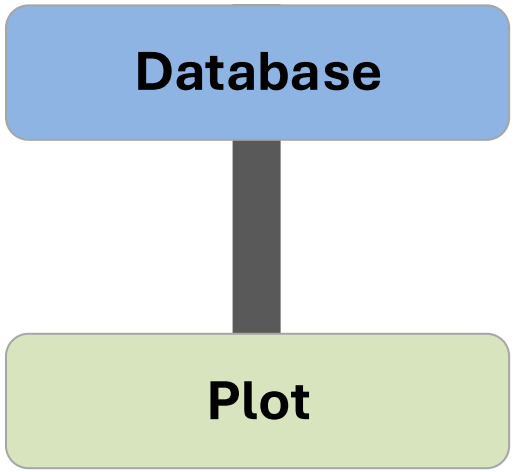
VisIt's Visualization Building Blocks

VisIt's interface is built around five core abstractions

- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data

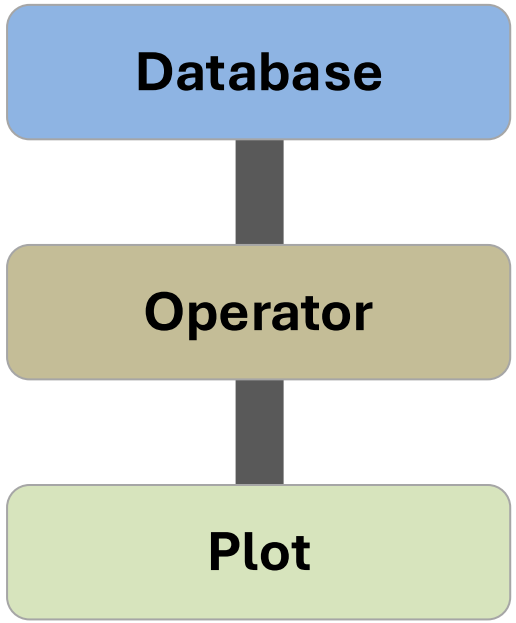


Open a database, which reads from a file
(Example: Open file1.hdf5)

Make a plot of a field in the database
(Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



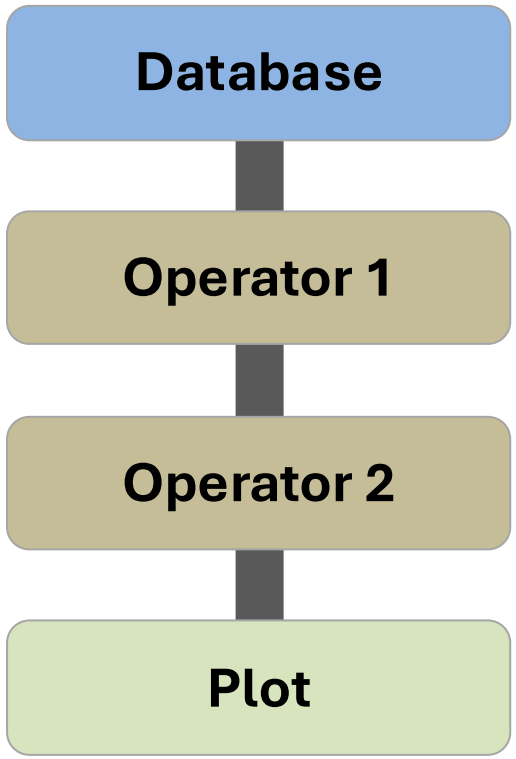
Open a database, which reads from a file
(Example: Open file1.hdf5)

Apply an operator to transform the data
(Example: Slice operator)

Make a plot of the result
(Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



Open a database, which reads from a file (Example: Open file1.hdf5)

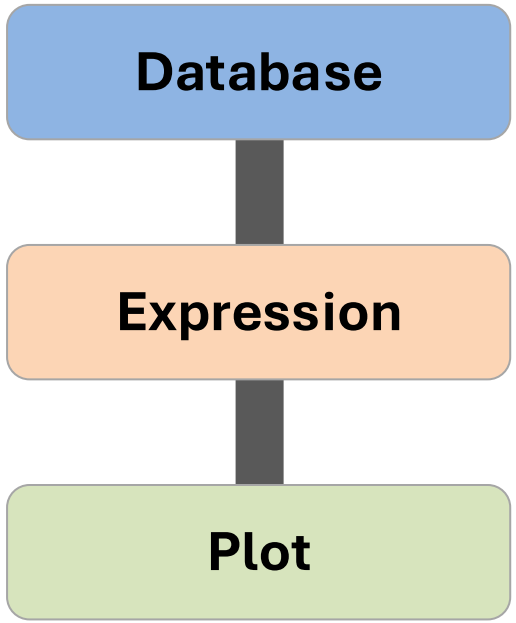
Apply an operator to transform the data (Example: Slice operator)

Apply a second operator to transform the data (Example: Elevate operator)

Make a plot of the result (Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



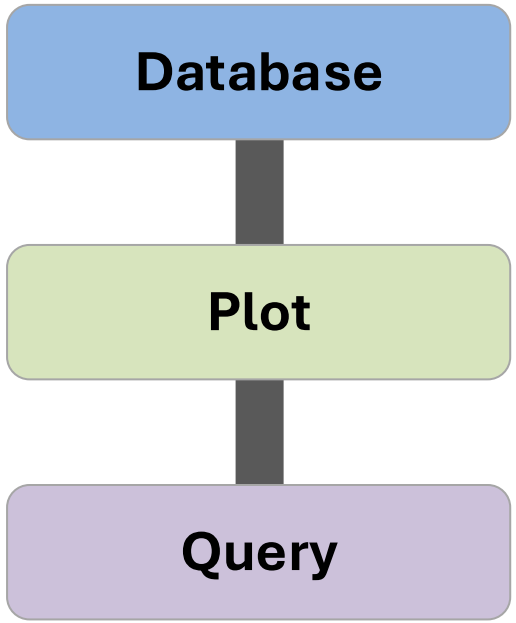
Open a database, which reads from a file
(Example: Open file1.hdf5)

Create derived quantity from existing fields
(Example: *speed* = *magnitude(velocity)*

Make a plot of the result
(Example: Pseudocolor plot of *speed*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



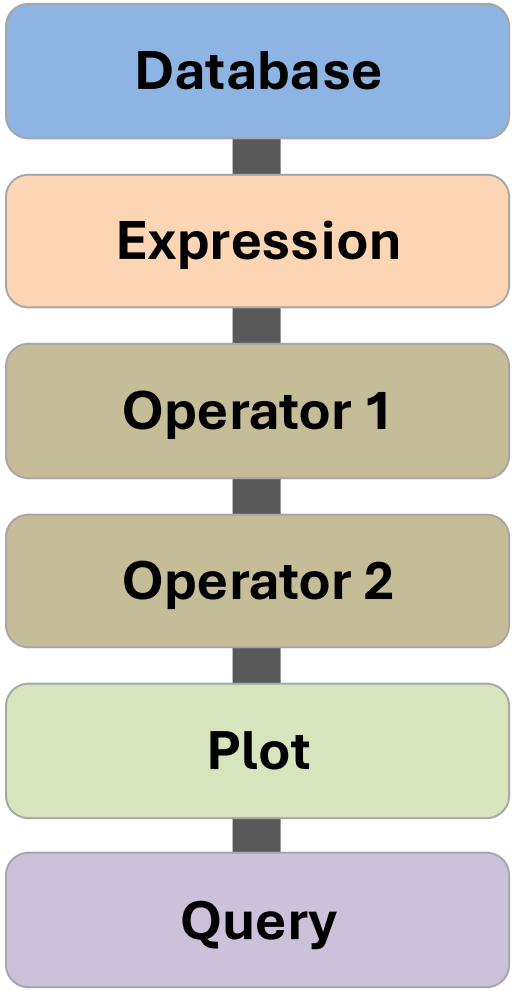
Open a database, which reads from a file
(Example: Open file1.hdf5)

Make a plot of a field in the database
(Example: Pseudocolor plot of *density*)

Extract quantitative information
(Example: integrate *density* to find *mass*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



Open a database, which reads from a file (Example: Open file1.hdf5)

Create derived quantity from existing fields (Example: *speed* = *magnitude(velocity)*

Apply an operator to transform the data (Example: Slice operator)

Apply a second operator to transform the data (Example: Elevate operator)

Plot a field (Example: Pseudocolor plot of *speed*)

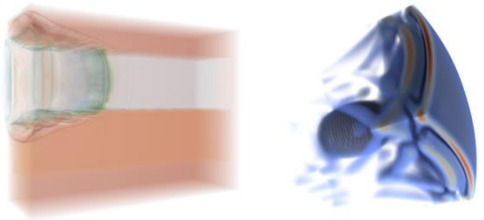
Extract quantitative information (Example: Maximum speed over cross-section)

Ascent Project Introduction

Ascent is an easy-to-use flyweight in situ visualization and analysis library for HPC simulations

- **Easy to use in-memory visualization and analysis**
  - Use cases: ***Making Pictures, Transforming Data,*** and ***Capturing Data***
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
  - Efficient distributed-memory (MPI) and many-core (CUDA, HIP, OpenMP) execution
    - Demonstrated scaling:
      In situ filtering and ray tracing across ***16,384 GPUs*** on LLNL's Sierra Cluster
  - Has lower memory requirements than current tools
  - Requires fewer dependencies than current tools (ex: no OpenGL)

Ascent



Visualizations created using Ascent

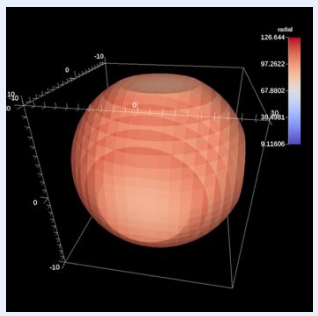
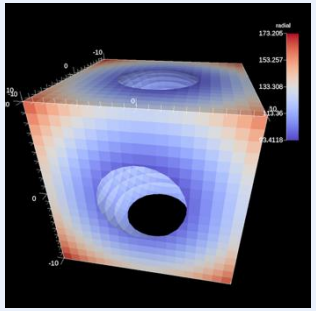
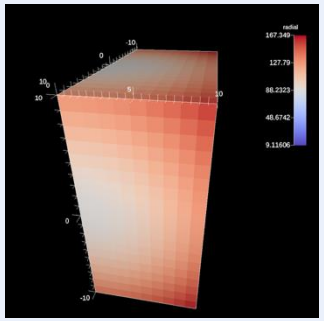
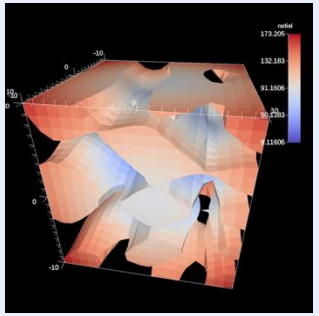
HDF python™ CINEMA

Extracts supported by Ascent

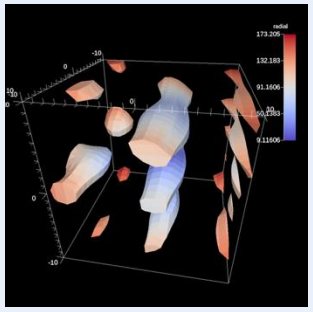
http://ascent-dav.org
https://github.com/Alpine-DAV/ascent

Website and GitHub Repo

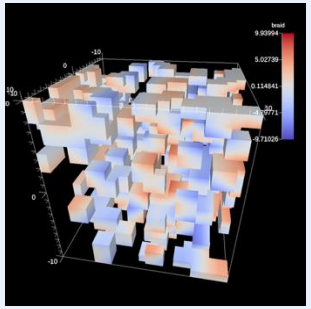
Ascent supports common visualization use cases

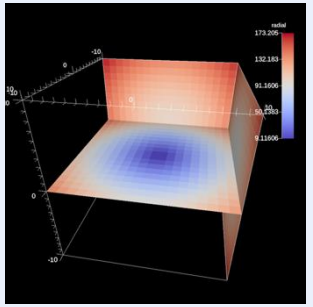
Clips



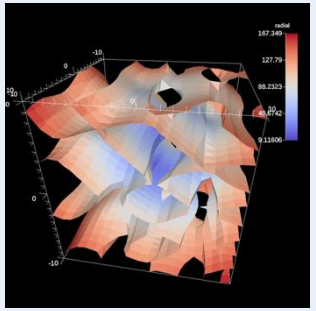
Iso-Volume



Threshold



Slice



Contour

[powered by]

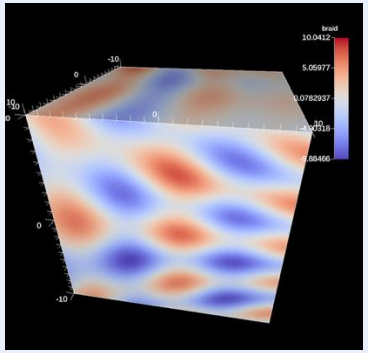
VISKORES

RAJA

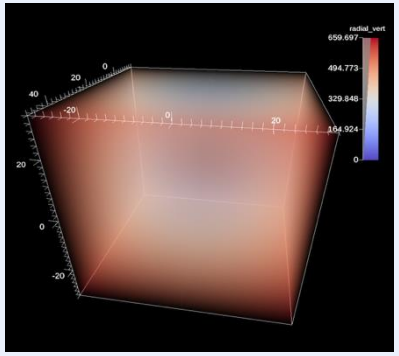
mfem

Devil Ray

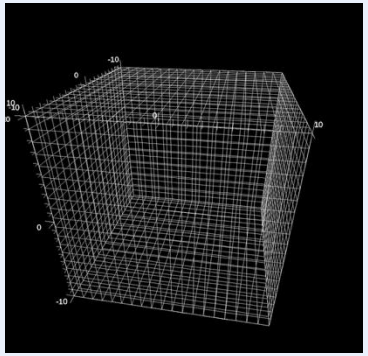
Rendering



Pseudocolor



Volume

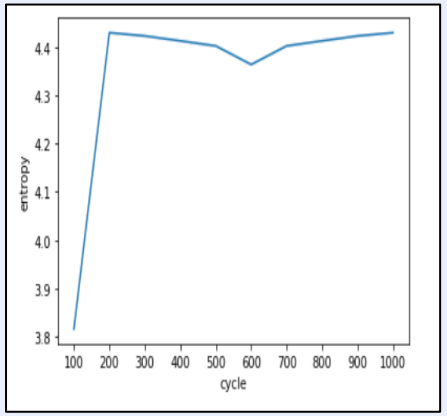


Mesh

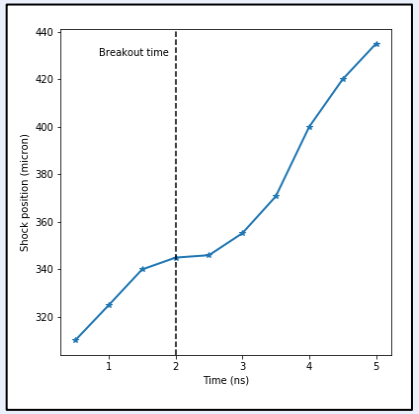
Ascent supports common analysis use cases

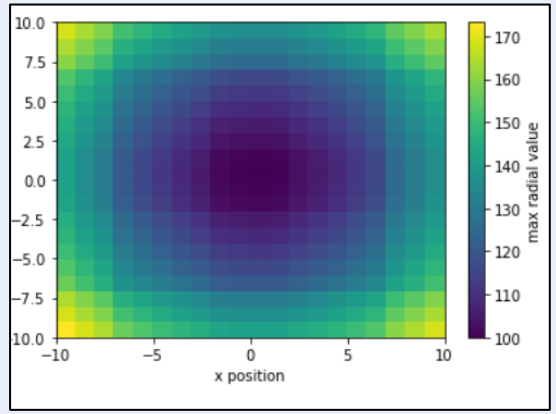
```
expression: |
  du = gradient(field('velocity','u'))
  dv = gradient(field('velocity','v'))
  dw = gradient(field('velocity','w'))
  w_x = dw.y - dv.z
  w_y = dw.z - dv.x
  w_z = dw.x - dv.y
  vector(w_x,w_y,w_z)
name: vorticity
```

Derived Fields



Time Histories





Lineouts and Spatial Binning

```
condition:
  entropy - history(entropy,
                    relative_index = 1) > 0.5
```

Triggers

Extracts

Scalar Images

HDF5 Files

Cinema Databases

Ascent is being used at scale on the DOE's exascale supercomputers

- Ascent is being used for rendering and data reduction on El Captain
  - Ascent extracts and external surfaces extracts are enabling post-hoc movies using VisIt
  - High demand and high hopes for in situ data reduction to support AI/ML efforts
- Ascent was used with NASA FUN3D INCITE runs on OLCF's Frontier and will be used as part of their ALCF Aurora simulations

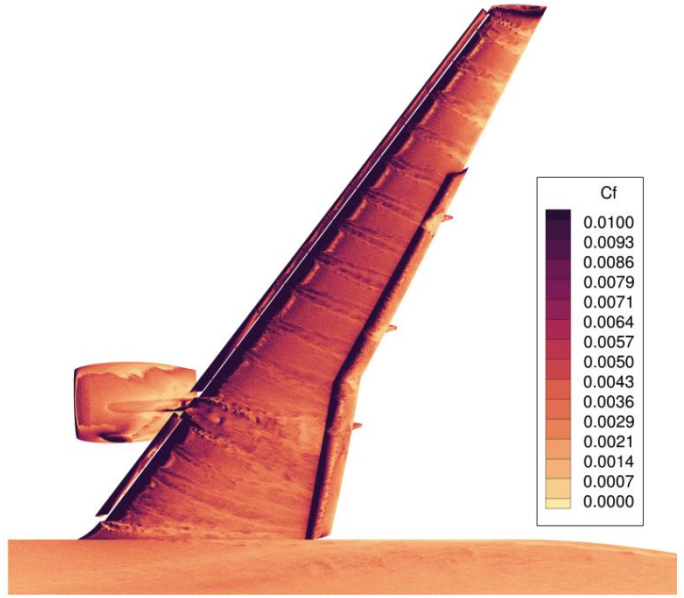


Fig. 12 Wing planform view showing contours of the skin friction magnitude.

Image from Mark Lohry, NASA Langley

Nielsen, E.J., Walden, A., Nastac, G., Wang, L., Jones, W., Lohry, M., Anderson, W.K., Diskin, B., Liu, Y., Rumsey, C.L. and Iyer, P., 2024. Large-Scale Computational Fluid Dynamics Simulations of Aerospace Configurations on the Frontier Exascale System. In *AIAA AVIATION FORUM AND ASCEND 2024* (p. 3866) https://doi.org/10.2514/6.2024-3866

This success is the result of 10+ years of development since the Strawman Viz Proxy App. Ascent owes its success to extensive work across the HPC ecosystem in key libraries: VTK-m/Viskores, Conduit, Devil Ray, MFEM, RAJA, Umpire, and Kokkos.

We released Ascent 0.9.5 (Sept 25), 0.9.4 (July 25)

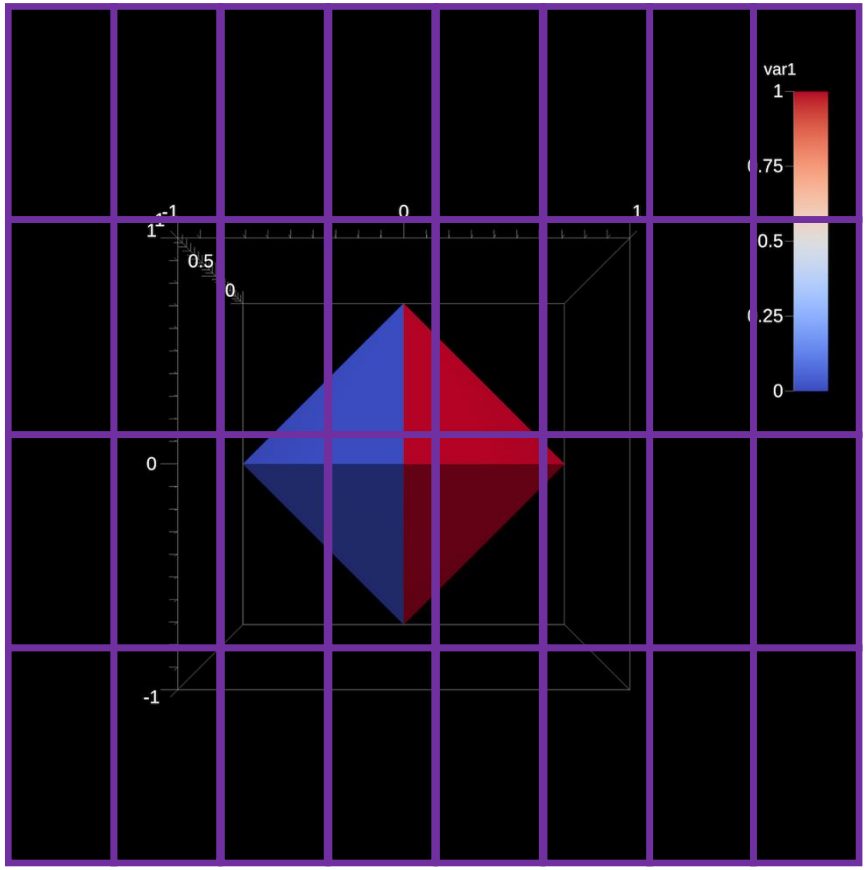
Highlights:

- VTK-m 2.3 support, Kokkos 4.7.00 support, and Conduit 0.9.5 support
- Extracts: Adding ZFP HDF5 options and adding Silo
- New logging and performance annotation infrastructure & more runtime diagnostic output
- Adding 2d camera view modes for project_2d scalar renderer
- Affine transform filter to rotate, scale, reflect and translate mesh coordinates
- Improvements to default scene cameras
- Added 1D & 2D contour capabilities
- Improvements to project_2d, uniform grid sampling, slicing, and simulated radiography filters.
- Added camera frustum information of rendered images to Ascent::info()
- Adding support for unstructured topologies with mixed element types
- New external surfaces and point-based sampling filters
- Unified file name formatting options
- Rover: support for Absorption-only and optical depth compositing
- Caching functionality for the actions

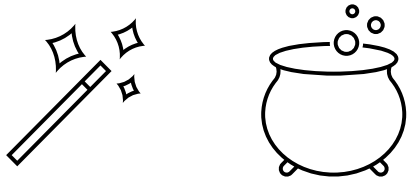
We will Ascent 0.9.6 in late summer 2026

Highlights:

- New tiled mode for high-resolution rendering
- New distributed-memory sampling modes:
  - Implicit points, lines, planes and boxes
  - Explicit point lists
  - Topology-based
    (arbitrary surfaces and volumes)
- New arbitrary ray-based field sampling
  - An extension of scalar rendering
- Refactored internal parameter validation to use a `JSON Schema` approach
  - Enables auto generated UIs and docs
  - Machine-readable API schemas facilitate AI-assisted workflows



Tiled Rendering Example
8192 x8192 image rendered using 32 tiles of size 512x1024



Ascent's API

Ascent's interface provides five top-level functions

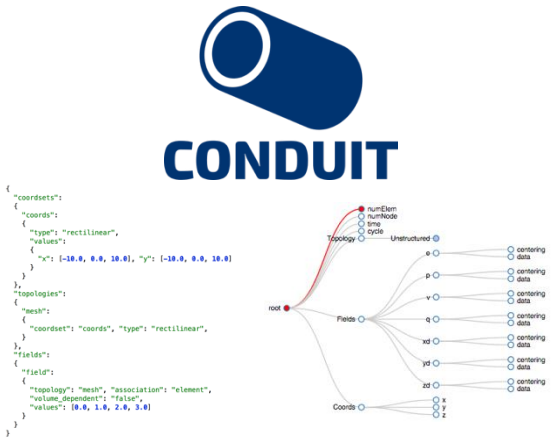
- **open() / close()**
  - Initialize and finalize an Ascent instance
- **publish()**
  - Pass your simulation data to Ascent
- **execute()**
  - Tell Ascent what to do
- **info()**
  - Ask for details about Ascent's last operation

```
//
// Run Ascent
//

Ascent ascent;
ascent.open();

ascent.publish(data);
ascent.execute(actions);
ascent.info(details);

ascent.close();
```

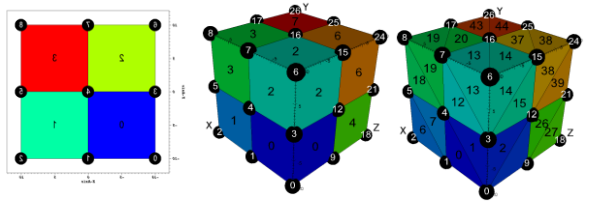
The *publish(), execute(),* and *info()* methods take Conduit trees as an argument.

Conduit provides intuitive APIs for in-memory data description and exchange

- **Provides an intuitive API for in-memory data description**
  - Enables *human-friendly* hierarchical data organization
  - Can describe in-memory arrays without copying
  - Provides C++, C, Python, and Fortran APIs
- **Provides common conventions for exchanging complex data**
  - Shared conventions for passing complex data (e.g. *Simulation Meshes*) enable modular interfaces across software libraries and simulation applications
- **Provides easy to use I/O interfaces for moving and storing data**
  - Enables use cases like binary checkpoint restart
  - Supports moving complex data with MPI (serialization)



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

http://software.llnl.gov/conduit
http://github.com/llnl/conduit

Website and GitHub Repo

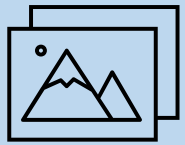
Ascent uses Conduit to provide a flexible and extendable API

- Conduit underpins Ascent's support for C++, C, Python, and Fortran interfaces
- Conduit also enables using YAML to specify Ascent actions
- Conduit's zero-copy features help couple existing simulation data structures
- Conduit Blueprint provides a standard for how to present simulation meshes

Learning Ascent equates to learning how to construct and pass Conduit trees that encode your data and your expectations.

Ascent's interface provides five composable building blocks to users

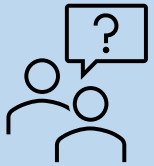
Scenes
(Render Pictures)



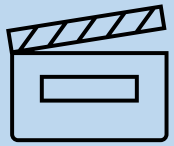
Pipelines
(Transform Data)

Extracts
(Capture Data)

Queries
(Ask Questions)



Triggers
(Direct Actions)



Ascent's Tutorial Examples demonstrate these building blocks

Contact info and Resources

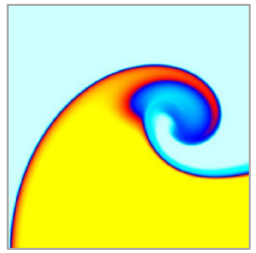
Presenter Contact Info:

- Cyrus Harrison: cyrush@llnl.gov

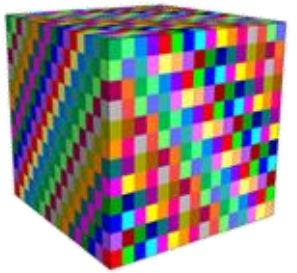
Resources:

- VisIt
  - Website: http://www.llnl.gov/visit
  - Github: https://github.com/visit-dav/visit
  - GitHub Discussions: https://github.com/visit-dav/visit/discussions
- Ascent
  - Website: http://www.ascent-dav.org
  - Github: https://github.com/alpine-dav/ascent

Bonus Content

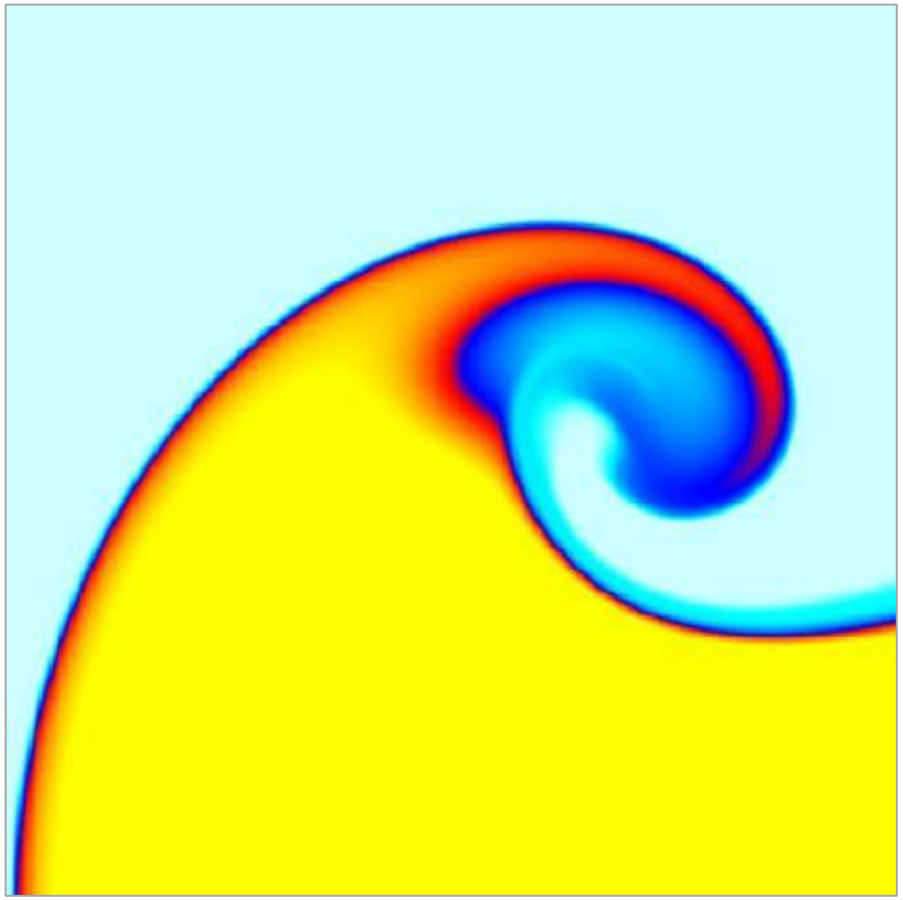


Visualization Techniques for Mesh-based Simulations



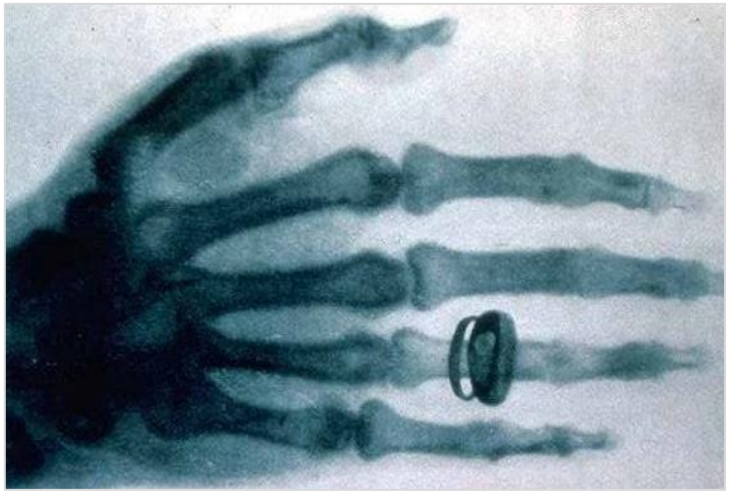
Pseudocolor rendering maps scalar fields to a range of colors

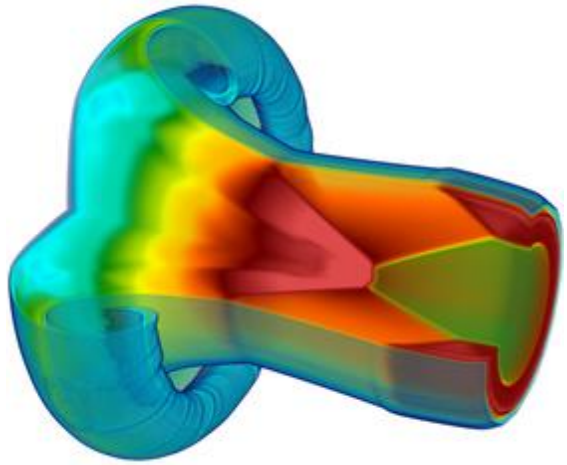
Pseudocolor rendering of Elevation

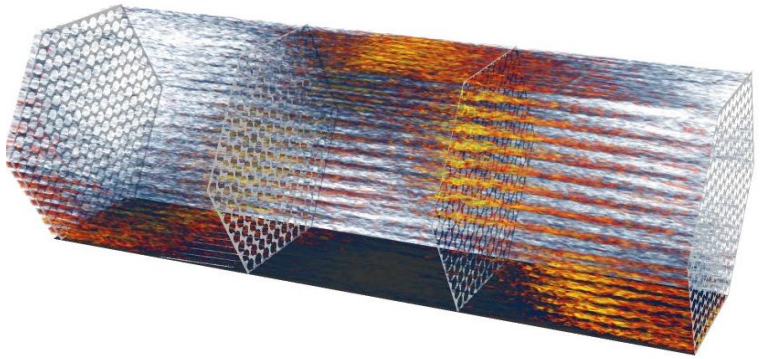


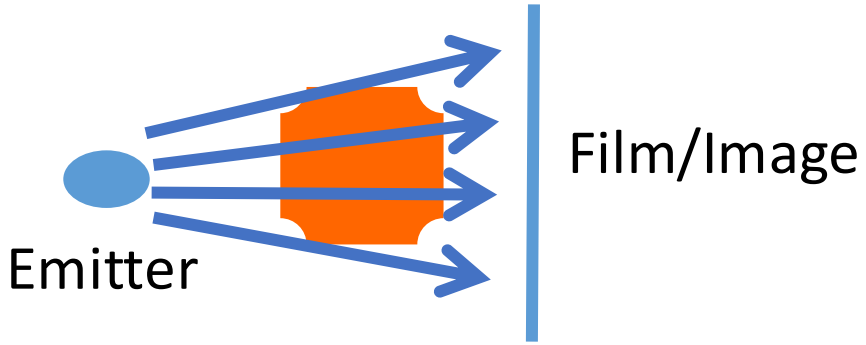
Pseudocolor rendering of Density

Volume Rendering cast rays though data and applies transfer functions to produce an image

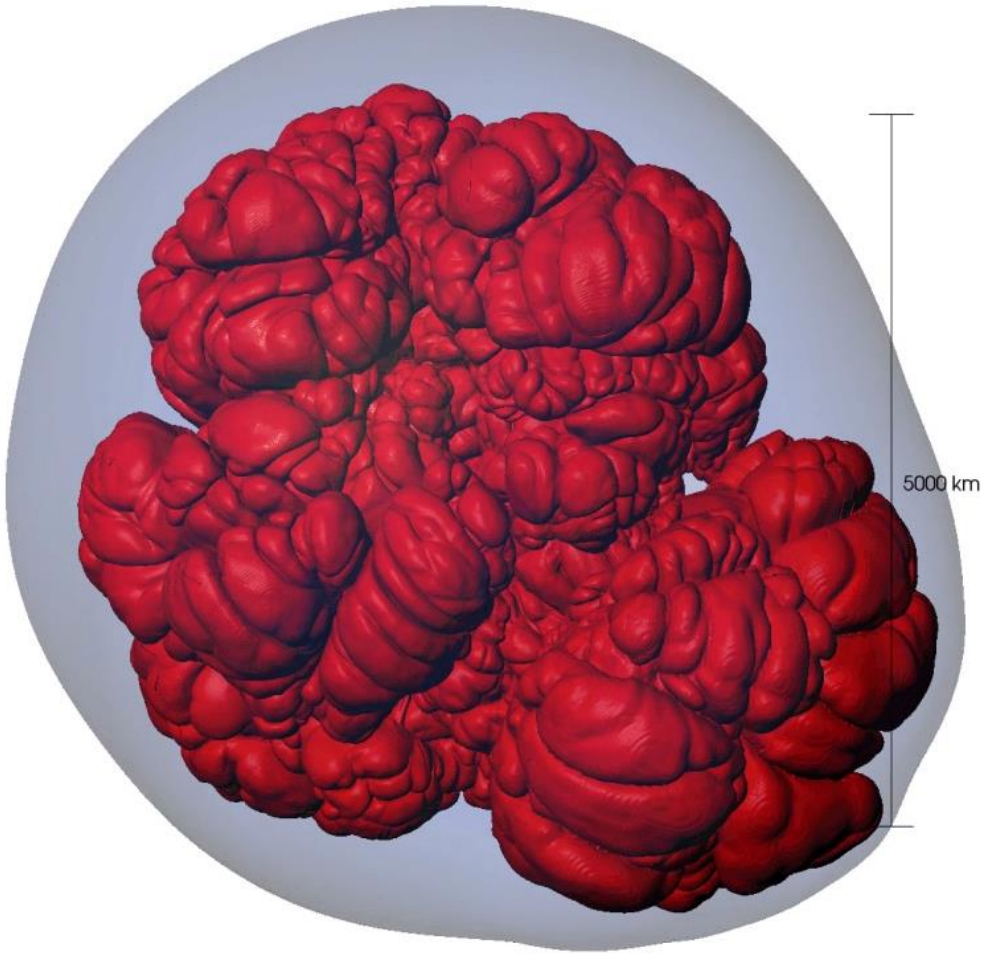








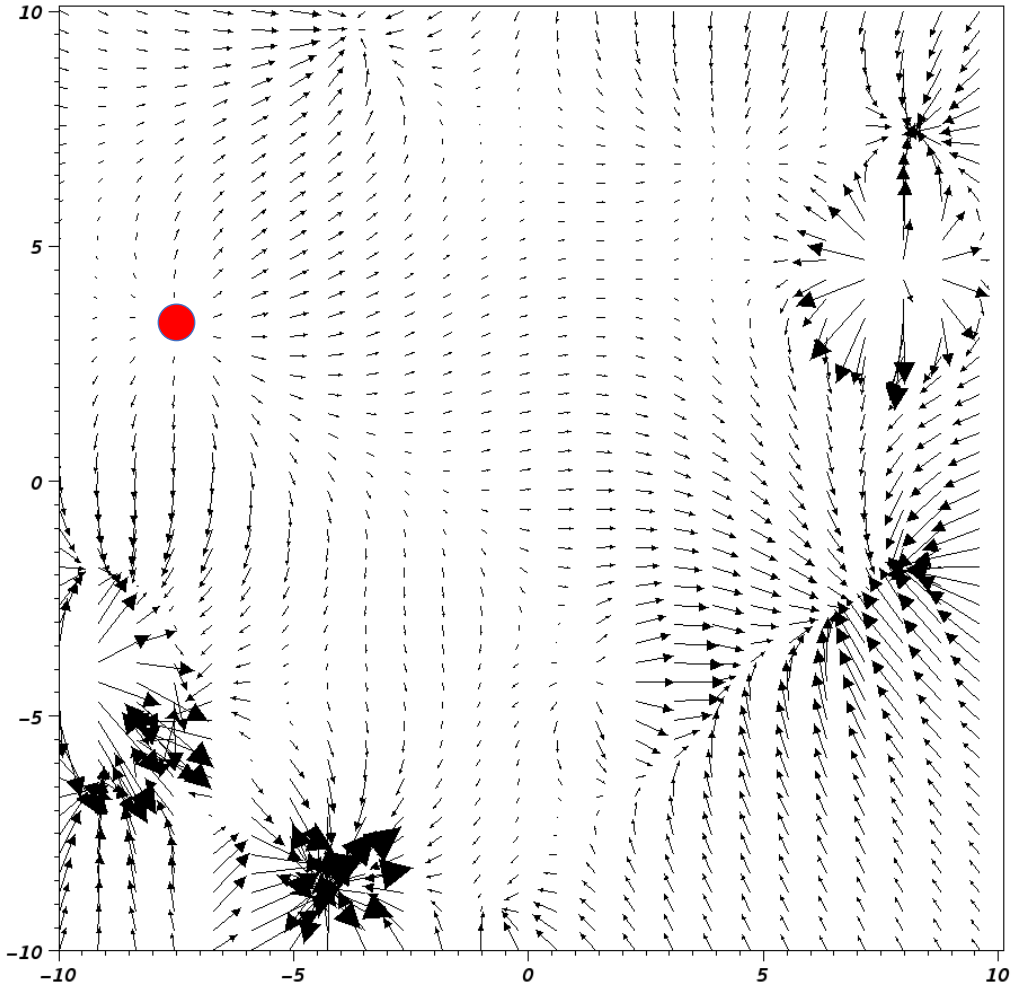
Isosurfacing (Contouring) extracts surfaces of that represent level sets of field values



Particle advection is the foundation of several flow visualization techniques

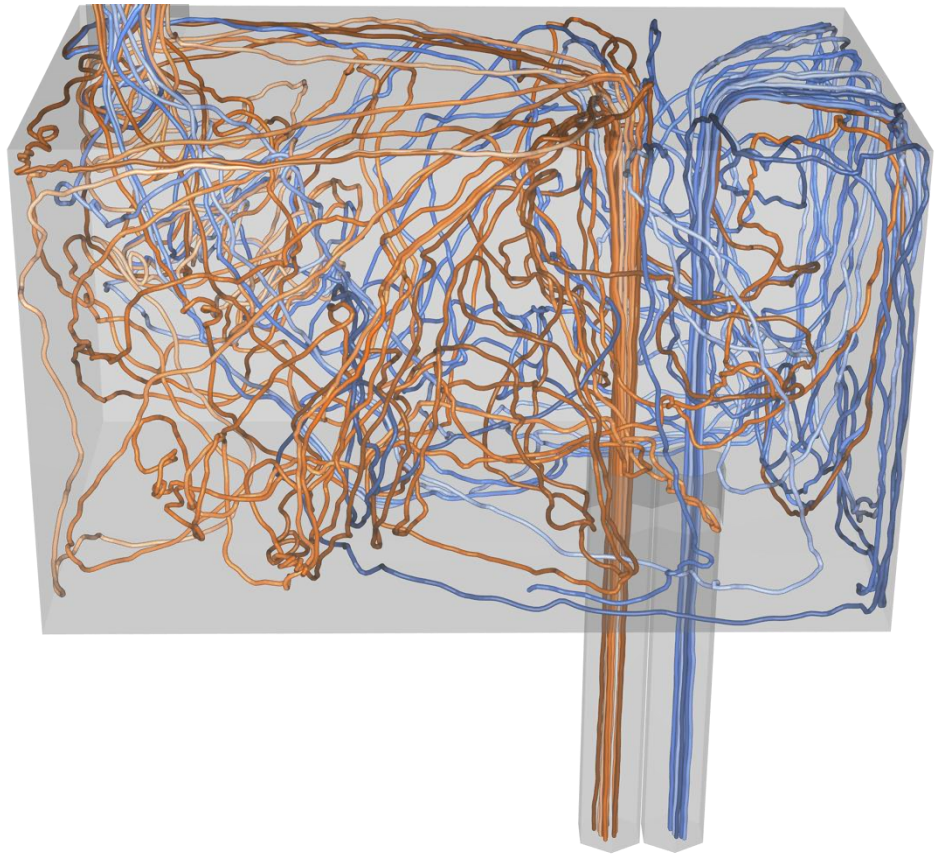
- S(t) = position of particle at time t
- $S(t_0) = p_0$
  - t_0: initial time
  - p_0: initial position
- S’(t) = v(t, S(t))
  - v(t, p): velocity at time t and position p
  - S’(t): derivative of the integral curve at time t

This is an ordinary differential equation.



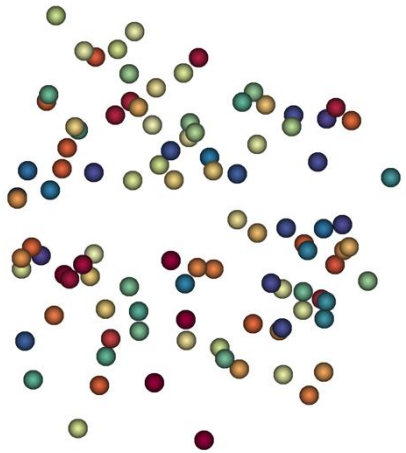
Streamline and Pathline computation are built on particle advection

- **Streamlines** – Instantaneous paths
- **Pathlines** – Time dependent paths

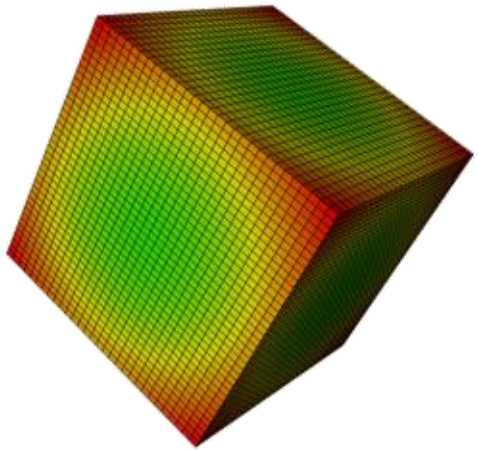


Meshes discretize continuous space

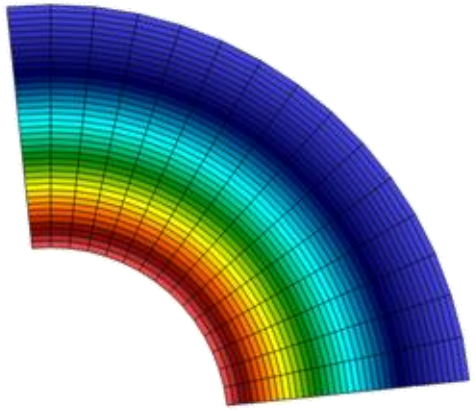
- **Simulations use a wide range of mesh types, defined in terms of:**
  - A set of coordinates (“nodes” / “points” / “vertices”)
  - A collection of “zones” / “cells” / “elements” on the coordinate set



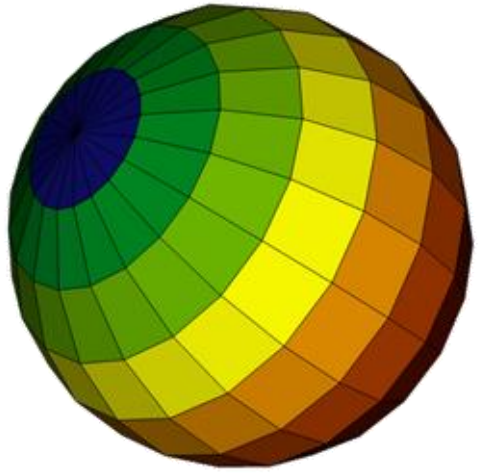
Points



Uniform



Curvilinear

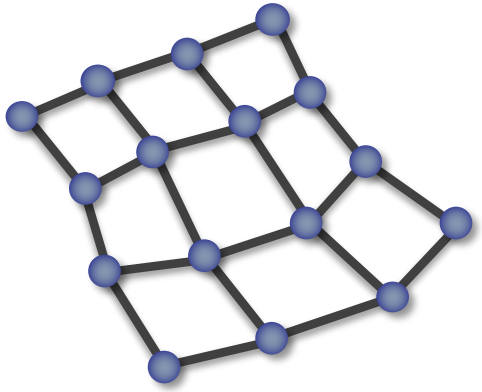


Unstructured

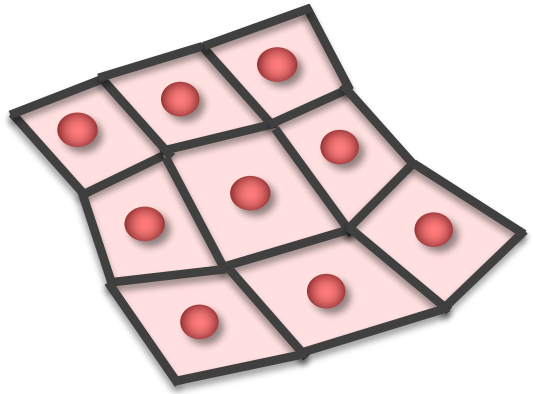
VisIt uses the “Zone” and “Node” nomenclature throughout its interface.

Mesh fields are variables associated with the mesh that hold simulation state

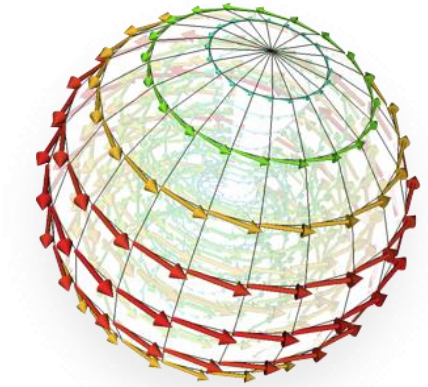
- Field values are associated with the zones or nodes of a mesh
  - Nodal: Linearly interpolated between the nodes of a zone
  - Zonal: Piecewise Constant across a zone
- Field values for each zone or node can be scalar, or multi-valued (vectors, tensors, etc.)



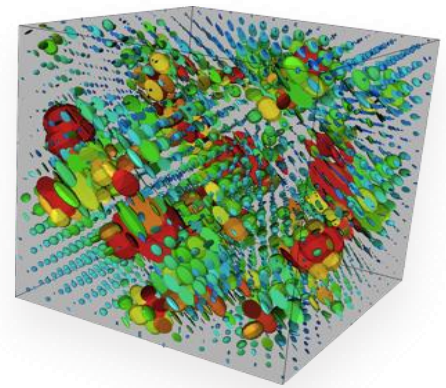
Nodal Association



Zonal Association



Vector Field

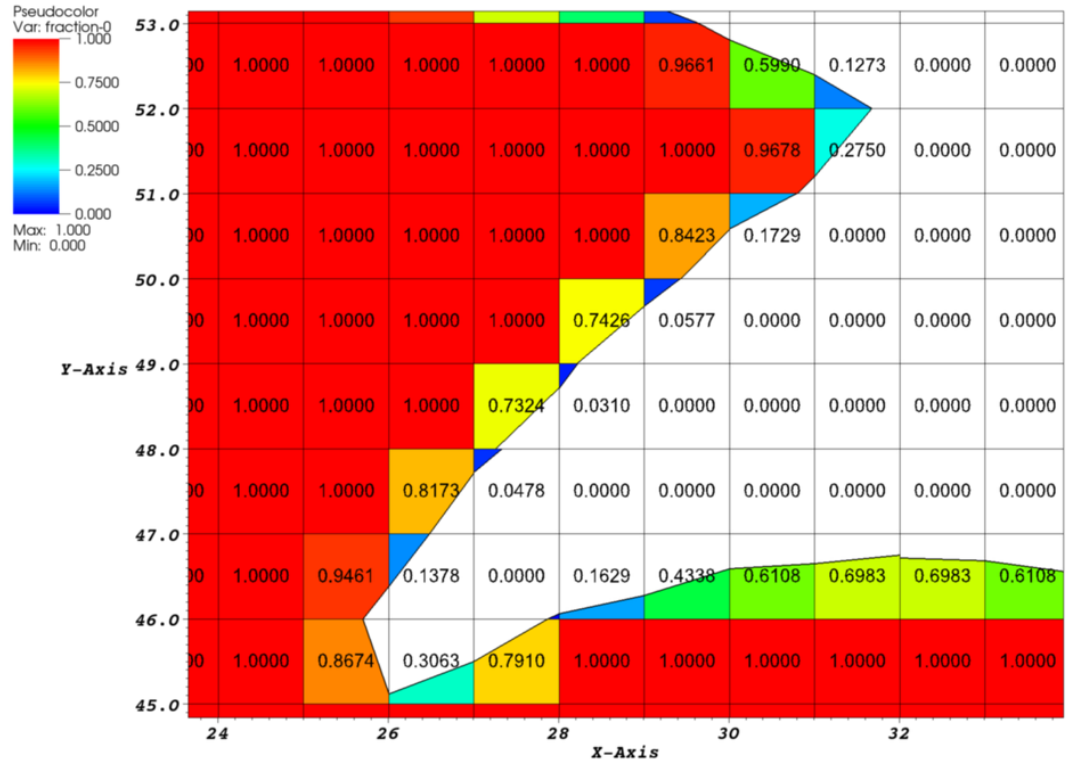


Tensor Field

Material volume fractions are used to capture sub-zonal interfaces

- Multi-material simulations use volume/area fractions to capture disjoint spatial regions at a sub-grid level.

- These fractions can be used as input to high-quality sub-grid material interface reconstruction algorithms.

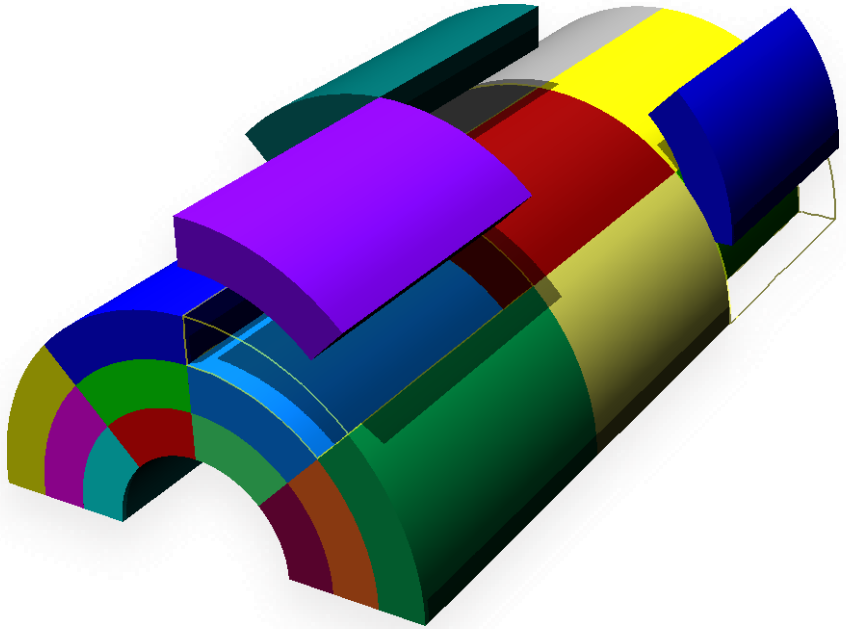


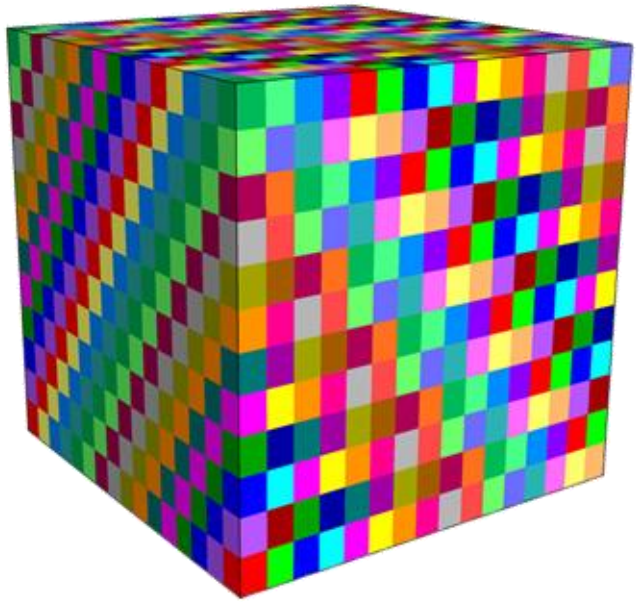
Species are used to capture sub-zonal weightings

- Species describe sub-grid variable composition
  - Example: Material “Air” is made of species “N2” ,“O2”, “Ar”, “CO2”, etc.
- Species are used for weighting, not to indicate sub-zonal interfaces.
  - They are typically used to capture fractions of “atomically mixed” values.

Domain decomposed meshes enable scalable parallel visualization and analysis algorithms

- Simulation meshes may be composed of smaller mesh “blocks” or “domains”.
- Domains are partitioned across MPI tasks for processing.





Adaptive Mesh Refinement (AMR) refines meshes into patches that capture details across length scales

- Mesh domains are associated with patches and levels
- Patches are nested to form an AMR hierarchy